Winning through incremental innovation: The case of MySQL AB

Jesper Holck[1], Volker Mahnke[2], and Roberto Zicari[3]

Abstract: *This empirical paper explores how entrants win substantial market acceptance, based on a series of incremental software innovations supported by an open source software development process, and a unique web-based proprietary business model. We investigate the competitive processes a case company (MySQL AB) employed to establish a sustainable foothold in the database software market - against the odds of a large installed customer base of competitors, and market incumbents' deep pockets and control of substantial complementary assets. Our findings suggest that a series of rapid incremental innovations, stack-based complement strategies, an open-source product testing approach, and a dual licensing strategy all contributed to winning through incremental innovation in increasingly commoditized database software markets. We distill seven important lessons that serve to guide managers in pursuing winning business models based on a series of incremental, and simultaneously pursued innovations.*

Keywords: *Incremental innovation, market share, open source, databases.*

1. Introduction

Many current theoretical and practical accounts of entrants winning markets through innovation are predicated on the crucial assumption that it takes radical, disruptive, or architectural, rather than incremental innovation to conquer substantial market share from incumbents, that enjoy network externalities, a substantial installed customer base, control complementary product markets, and command deep pockets (Utterback & Abernathy, 1975; Teece, 1986; Abernathy & Clark, 1985; Henderson & Clark, 1990). Indeed, in many industrial contexts, incumbents benefit from incremental innovation. Because of their large installed customer base, incumbents enjoy relatively greater return from incremental innovation, and also benefit from learning curve effects. If supply-side economies of scale are present, having more customers usually translates into greater cost efficiency, that may again be translated into higher margins and/or lower product prices.

At the same time, incremental innovations are usually thought to enhance an incumbent's capabilities, while radical ones are thought to undermine it. Finally, a large installed base makes providers of complementary products more likely to provide seamless interfaces to a focal incumbent's product offering. For all these reasons it appears plausible that winning through incremental innovation against a well-versed set of incumbents is difficult, if possible at all.

However, important exceptions appear to exist. Based on a thorough and longitudinal case analysis (Eisenhardt, 1989), we identify crucial elements and conditions that serve to develop an integrative model of winning through incremental innovation in

[1] Copenhagen Business School, Department of Informatics, Howitzvej 60, DK-2000 Frederiksberg, Denmark. jeh.inf@cbs.dk

[2] Copenhagen Business School, Department of Informatics, Howitzvej 60, DK-2000 Frederiksberg, Denmark. vm.inf@cbs.dk

[3] Universität Frankfurt am Main, Institute of Computer Science, Robert-Mayer-Str. 10, D-60325, Frankfurt /M., Germany. zicari@cs.uni-frankfurt.de

highly competitive markets, against powerful and well established incumbents. While current innovation theory suggests that winning through incremental innovation against incumbents is rare and difficult, we suggest that competitive processes employed by a case company – MySQL AB – serves as a powerful reminder that a sustainable foothold in the database software market could be established, based on a series of incremental and complementary innovation.

Despite all odds in the current market, our findings suggest that a combination of such incremental innovations has enabled competitive moves, including stack-based product development strategies, an open source product testing approach, and, importantly, dual licensing contracts. These moves all contributed to winning through incremental innovation in the increasingly commoditized database software markets against powerful competitors such as IBM, Oracle, and Microsoft.

The remainder of the paper is structured as follows. First, we briefly review current theories on innovative entry, which collectively suggest that winning through incremental innovation is an unlikely industry event. Secondly, we outline our research methodology and provide a techno-historical account on how MySQL AB entered the database market and won a sustainable foothold against dominant incumbents such as IBM, Microsoft, and Oracle. Third, we use the case findings to develop an integrative, yet parsimonious model of winning through incremental innovation. Fourth, we discuss alternative explanations of MySQL AB's success, and, finally, suggest fruitful avenues of future research.

2. Competitive effects of incremental innovation

Innovations are central in the explanation of competitive success – be they incremental or radical (Utterback & Abernathy, 1975; Teece, 1986; Abernathy & Clark, 1985; Henderson & Clark, 1990; Teece, Pisano, & Shuen, 1997). However, many theories predict that compared to new entrants, established incumbents will benefit most from incremental innovation[4].

Industry life-cycle theories of innovation (e.g. Utterback & Abernathy, 1975; Klepper, 1996) suggest that incumbent firms, engaging in process innovation during and after the emergence of a dominant design, excel in competition. Thus, in mature industrial contexts, incremental innovations are likely to benefit incumbent players most. Incumbents also enjoy greater returns from incremental innovation the greater their customer base (Klepper, 1996): returns are greater if more customers benefit.

In addition, a large installed customer base makes providers of complementary products more likely to offer seamless interfaces to a focal incumbent's product offering (Teece, 1982). This is particularly true if such complementarity effects are accompanied by network effects (Liebowitz & Margolis, 1998): while complementary products increase the demand of what an incumbent uniquely provides, network effects increase customer utility as more consumers adopt a focal firm's product (Dixit & Nalebuff, 1993)

Such supply-side scale effects work in combination with learning curve effects, further augmenting an incumbent's advantage (Nelson & Winter, 1982). While radical and/or architectural innovations threaten to undermine an incumbent's previous strength (Tushman & Anderson, 1986), incremental innovations usually enhance an incumbent's capabilities (Adner & Levinthal, 2001). The scale and learning effects in

[4] For a comprehensive summary see (Tran, 2005)

conjunction provide cost advantages that offer incumbents greater competitive pricing possibilities, compared to new industry entrants (Henderson & Clark, 1990).

As a result, incumbents in most industries usually win against new entrants if entrants compete on incremental innovation. It is therefore surprising, that winning through incremental innovation is indeed possible, as illustrated by the following case.

3. The case of MySQL AB

3.1 Methodology

We will in this section present a "techno-historical" account of the company MySQL and its software products, and use this description as basis for a subsequent analysis of MySQL's strategy. Based on this analysis, we will then discuss in what ways the case of MySQL's success can be understood in terms of the existing literature on incremental innovation, and finally suggest a few striking "lessons to be learned" from this case story.

3.2 MySQL: A techno-historical account

MySQL AB was in February 2008 acquired by Sun Microsystems, Inc., for close to $1 billion in total consideration. This event is, however, just the endpoint of an evolutionary wealth-creating process, made possible by a series of incremental innovation through which the firm has become a serious competitor and sustained a foothold alongside IBM, Microsoft, and Oracle, the big three IT companies in the global high-margin-commodity database business.

MySQL is a SQL database management system (DBMS) with more than 10 million installations, and approximately 50,000 daily downloads. The basic program runs as a server, providing multi-user access to a number of databases. Until Sun's acquisition, MySQL was under the ownership of a single for-profit firm, the Swedish company MySQL AB, which also holds the copyright to most of the code base. MySQL AB was founded by David Axmark, Allan Larsson, and Michael "Monty" Widenius. Mårten Mickos, the current CEO, joined later as the company grew and sought professional management.

1982-1995: Foundation, the first technological battle

MySQL AB's history goes back to 1982, when MySQL's co-founder Michael "Monty" Widenius while employed at TcX, a Swedish IT and data consulting firm, authored UNIREG – a program for administrating ISAM data stores from a text-based terminal.[5] UNIREG was originally written in BASIC for what was at that time state-of-the-art personal computers with 4 MHz Z80 processors and 16kB of RAM. In the following years UNIREG was ported to other platforms, from 1986 including Unix.

However, around 1994 UNIREG began to appear outdated, and TcX began to turn its attention to mSQL (mini SQL) – a proprietary, but inexpensive data base management system developed in 1994 by Hughes Technologies, which filled an unserved market need between desktop systems like Microsoft Access and enterprise-level systems like Oracle and DB2. While mSQL had the advantage of a well-developed client application program interface (API), its initial version was not compatible with UNIREG and also lacked indexing, a feature crucial for performance of large data storage systems.

[5] ISAM (Indexed Sequential Access Method), originally developed by IBM for mainframe computers, is a method for storing large amounts of data for fast retrieval. ISAM forms the basic data store for almost all databases.

So the idea of simply using mSQL as an alternative front-end for UNIREG was eventually abandoned.

Instead, TcX decided to develop a new product, combining the effective UNIREG indexing scheme with an API very similar to mSQL's. This product was released in 1995 under the name of MySQL version 1. By re-using mSQL's API, TcX made it easy to port applications from mSQL to MySQL. mSQL and MySQL were for some time competitors on the Linux platform, but MySQL eventually won this initial battle, due to advantages such as fast and efficient code, developed for very small and slow computers.

The first release of MySQL was not open source, instead TcX chose to release the software as "shrink wrap", but with a license that required no payment for deployment on Unix servers – a license similar to the one used by Aladdin Software for Ghostscript. The software was freely distributable, and commercial use was allowed as long as the software was not distributed commercially. Part of MySQL's initial success is no doubt based on the choice of a quite "open" license, compared to the one used for mSQL (Axmark & Widenius, 1999).

It is worth noting that relational databases (RDBMS[6]) – compared with most other IT technologies – is a quite old and well-established concept. The basic principles were developed by Codd as early as 1970 (Codd, 1970); SQL, the standard language for using relational databases, was first described in 1974, and later adopted as an ANSI standard in 1986 and as an ISO standard in 1987. So already in the 1980s, the RDBMS market was technologically quite mature in the sense that customers knew what to expect of a RDBMS and which competing products were relevant to consider.

1996-2000: Riding on neglected Web service market opportunities

When MySQL.com was formed to maintain the database, its investors requested that MySQL would be made open source, so they knew the product remained accessible for use, in case the company would run into trouble. MySQL AB released its product under the GPL (Gnu Public License).

From 1996 to 2006 MySQL gained an increasingly stable foothold in the database market by serving the needs of the growing Web service community, initially ignored by the market leaders (IBM, Microsoft, Oracle). While the initial target was on web-enabled companies (including Yahoo, Google, Wikipedia, and many smaller ones), after the foothold was established, MySQL was in the position potentially to challenge every player in the RDBMS market.

Nonetheless, MySQL had several draw-backs compared to competing products, not least lack of support for important SQL elements like stored procedures, transactions, and foreign keys. But in the late 1990s, where "everyone" joined the Internet and WWW revolution, these draw-backs were of less importance.

The company had already for a long time tried to follow the "15 minute rule", stating that it should be possible to get MySQL up and running in less than 15 minutes. This made MySQL a very attractive choice for Internet start-ups, not wanting to spend days or weeks on installing complicated RDBMSs and on participating in vendors' courses necessary to install and administer these systems.

Because most web-administrators had no formal education in computer systems and database technology, they were less considered with the missing features of MySQL. They needed the basic RDBMS functionality as a data store for customers, goods, etc.; unlike traditional enterprise RDBMS users, they were not too worried about e.g.

[6] Relational DataBase Management System

the small risk of an eventual inconsistency in the database, and hence did not feel a great need for support of ACID transactions.[7]

The free (both in the sense of gratis and the sense of freedom) availability of the MySQL software made it an attractive option for start-up companies joining the Internet revolution. Even for established companies, the free availability of MySQL could be a major advantage. For example, an interesting aspect of a gratis software in large organizations is that "nerds" in the computer department can test it of, e.g. for an experimental web site, without asking for financial resources to do this (Holck et al., 2005). If the software proves to be of good quality, it can then gradually spread in the organization.

Compared with legacy products like DB2 and Oracle, MySQL had the strong advantage of demanding very little computer resources. Even on inexpensive PCs, MySQL could run quite efficiently. Partly because of MySQL being open source, a lot of 3rd party applications (both open source and proprietary) became quickly available, e.g. for on-line discussion forums and e-commerce. Typically these products were based on Linux as the operating system; Apache as the web server; MySQL as the database; and Perl, Python or PHP as the scripting language. As this technology matured, it became known as the LAMP stack[8], and attracted lots of seamless 3rd party products and services, education, and supporting books. Regarding the LAMP stack, a manager explained to us:

> *"MySQL got ignited from the Perl community, through being a better substitute for mSQL (more stable, more performance). And MySQL grew with the PHP community, becoming and staying a default in most PHP scenarios. MySQL recently got a default status with Ruby on Rails. Programming language communities are hugely important, when it comes to growing the user base*".

Regarding MySQL's entry on the RDBMS market, he commented:

> "*MySQL initially focused on the web-based database market. In a way, that's niche. We conquered an under-served niche with a product aspiring to be general-purpose. Then, we've conquered related / neighbouring niches. We have not grown market share equally in all markets."*

In 1998, Linux Journal announces MySQL as the Readers Choice Award: Favorite Database, an award MySQL receives every year from 1998 to 2003. The 1999 announcement said:

> *"MySQL, the free database from Finland and Sweden, scores big points with 42.4%, compared to PostgreSQL with 19.5% and Oracle with 15.2%. [...] it stands to reason that MySQL would be a favorite choice of the Linux community, considering its origins and relatively free license."* (Kroll, 2000)

2001-2007: Gaining traction and creating value in increasingly contested terrain

In 2001, in the aftermath of the Internet bubble, MySQL AB consisted of 20 people focusing mainly on technology, but largely neglecting managerial issues. In order to strengthen the company, the founders recruited a number of highly professional people for the top management, including chairman of the board John Wattin and CEO Mårten Mickos. These appointments coincided with the incorporation of the company in Sweden as MySQL AB. Subsequently, the company showed a strong growth in

[7] ACID = Atomicity, Consistency, Isolation, Durability: a set of properties that guarantee reliable processing of database transactions.

[8] The term was apparently first mentioned under this acronym in an article from 1998 (Kunze, 1998).

employees, operating income, and profits as illustrated below (data from Bureau van Dijk's Amadeus database):

	2006	2002
Operating revenue (thousand SEK)	285,742	5,611
Total assets (thousand SEK)	253,934	28,486
Employees	285	32
Profit (loss) before tax (thousand SEK)	-115,897	-14,460

This strong growth went hand in hand with establishment of several international offices and subsidiaries, including US, Germany, France, UK, Ireland, Benelux, Japan, and Italy. MySQL also established partnerships with a large number of companies, including NEC America, Sabre, Novell, Cox Communication, SAP, QNX, IBM, Alcatel, German Lotto, Sage Group, and Unisys. More professional people were headhunted, e.g. Kaj Arnö (VP of community relations, 2005). In order to fuel this growth, venture capital was raised in 2001, 2003 ($19.5 million), and 2006 ($18.5 million).

An important and interesting issue in these years was the "battle of the storage engines". The MySQL database must be supported by a storage engine, the program that handles how data are read from and written to the files etc. where the data are stored. The most common and fastest storage engine for MySQL has probably been MyISAM. For applications with high demands on concurrency, row-level locking, and transaction support (ACID compliance), InnoDB has been the preferred choice. InnoDB is included in the MySQL Enterprise Server and customers wanting to use MySQL and InnoDB under a commercial license can purchase this through MySQL. In an interesting move, Oracle Corporation in October 2005 bought Innobase OY, the company behind InnoDB. Later in February 2006, Oracle increased the pressure on MySQL, as it also purchased Sleepycat Software, distributor of Berkeley DB, an alternative storage engine.

As both InnoDB and Berkeley DB are open source software, the fact that Oracle Corporation purchased Innobase and Sleepycat did not appear an immediate threat to MySQL's business model, but MySQL felt the heat emanating from increasing competitors' attention. In response, the company engaged with Jim Starkey, creator of InterBase, a database released by Borland as open source in 2000. In 2007 MySQL released its first "own" open source storage engine: Falcon.

MySQL was in February 2008 acquired by Sun Microsystems, Inc., for close to $1 billion in total consideration. This event marked an endpoint in the evolutionary wealth-creating process, made possible by a series of incremental innovation

4. MySQL AB Incremental Innovative Strategy

The development of MySQL does not seem to introduce major technical innovation; only small incremental improvements to the basic technology were repeatedly implemented. These and the following seven "lessons" for winning through incremental innovation can be extracted. Much of MySQL's effort seems to be have put into making "behind the scenes" improvements like optimizations, data compression etc.

Lesson #1: Piggyback on existing standards

Always following existing standards and interfaces. By relying on the existing API of mSQL and the existing ISAM data storage technology, MySQL AB made it easy for users of these technologies to switch to MySQL. A firm commitment to existing standards reduces switching costs and helps customers migrating from other suppliers. Do not let technological advances make your product incompatible with existing software.

Lesson #2: Distinguish nice-to-have from must-have-features

Always prioritize a lean product (e.g. fast, easy to install, small memory footprint) over advanced nice-to-have features. Compared with both commercial and open source competitors (Oracle, PostgreSQL etc.), MySQL has been quite slow to adopt features like transactions and foreign keys, otherwise considered "must-have" for professional databases. However, as David Axmark and "Monty" Widenius suggest: "When designing MySQL, we had a greater need for speed than for transactions. It's no use having transactions, if the SQL server becomes so slow it's unusable for what you need to get done. There are, of course, applications that require transactions to work, but a wide range of applications work very well without them" (Axmark & Widenius, 1999)

Lesson #3: Seek supply-side economics of scale by coupling development of commercial and non-commercial product versions

MySQL chose to let both the Community and the Enterprise version build on the same code base, i.e. the exact same source code files. In Kaj Arnö's blog from October 17, 2006, where he introduces the MySQL Enterprise version, he only mentions *one* difference between the software offered to the two groups, namely that "we will do more frequent binary releases of the MySQL Enterprise Server software than of the MySQL Community Server", essentially just making it more convenient for Enterprise users to install the newest version of the software. Also, when a user in the MySQL forums raised the question of the difference between the Enterprise and the Community versions, Victoria Reznichenko, support engineer at MySQL answered:

> *"They are mostl[y] identical. To the recent Community release was added new feature SHOW PROFILE that is not included to Enterprise... InnoDB source is the same in both Enterprise and Community, but Enterprise builds are made more frequently. So if you hit a bug that [is] already fixed with Community binaries you need either to wait till the next Community release or build MySQL from source. With Enterprise subscription [y]ou get also access to MySQL support and support engineers will help you to find a cause of your problem."*

By linking the development of commercial and non-commercial product versions, economies of scale are realized. This coincides with the high importance that many open source projects make to maintain *one* common code base and avoid "forking"; if the code base is split into parallel streams, it quickly becomes very difficult to transfer bug-fixes and other code-changes from one stream to another (Holck & Jørgensen, 2003).

Lesson #4: Nurture and coordinate user communities

MySQL makes the following differentiation between the various groups involved in development and use of MySQL:

- MySQL *employees*: core developers, often selected from the open source developer community;
- *developer community*: active users, contributing with e.g. code, interfaces, forum answers, blog entries;

- *user community*: including inactive users, silently using the MySQL software.

An important factor for MySQL's success is the company's ability to create a single, *coherent* community out of the various groups with interests in the product: the free open source developer community, the community of developers at commercial customers, the company's core developers, etc. All of these developers collaborate using the Internet in a *virtual, coherent community associated with the product and its technology*. The feeling of belonging to this specific community is an important factor motivating members to take an active role in product development and marketing.

Lesson #5: Use communities to reduce product testing and marketing costs

It is important to note that most members of MySQL's open source software developer community perform an important support function, but do not generate revenue directly. With 50,000 product downloads per day and over 10 million active installations, most community members, even if they are non-paying users, help building a large community of marketeers and co-developers. While one commercial customer or even one commercial deal may correspond to hundreds of thousands, or millions, of installations; open source communities do contribute to reduce product testing and marketing costs. Community users, e.g., provide very valuable testing, error-reporting, and bug-fixing. Indeed, if MySQL chose to make new versions available for enterprise customers only, these versions would most likely show more errors, be less stable, and reach acceptable reliability levels later. In addition, because professional users can toy and play for free with non-commercial versions without asking for closely guarded IT budgets, they can and often do act as active bridgeheads into interesting customer accounts (Holck et al., 2005).

Lesson #6: Implement dual-licensing strategy

In contrast to many open source models, MySQL never ceded property rights to developed software code. So software contributions (e.g. bug-fixes) will only be accepted if the contributor accepts to transfer his/her copyright of the source code contribution to MySQL. As not all external contributors may want to do this, MySQL runs the risk of losing contributions that might significantly improve the software's quality.

But having sole copyright over the source code is a fundamental and necessary prerequisite for applying the dual-licensing strategy, where MySQL offers different licensing arrangements for different user groups: GPL[9] for non-commercial users; commercial license for commercial users. Dual licensing is an important element in benefiting from an incremental innovation based business model for software products that are embedded in other products. According to Mickos, most of MySQL AB's revenue is generated from the 0.5% who pay for embedding MySQL in their proprietary products.

We can identify two categories of paying customers: (a) Enterprise users, paying for the services included in the MySQL Enterprise subscription; and (b) Re-distributors buying the software with a non-GPL license and embedding it into their own, commercial product(s). For enterprise users, the dual licensing model is essentially equivalent to a support model – customers pay for licenses and in return receive support from MySQL AB. The revenue generated by MySQL Enterprise Subscription[10] is steadily increasing, which can be seen as an indicator that a large community may be a good base for revenue generation. One factor to consider is a possible large time lag

[9] Gnu Public License, the most popular open source license, implying that all software product derivatives must also be licenses under GPL (http://www.gnu.org/licenses/, accessed June 2008)

[10] MySQL at present offers four levels of Enterprise Subscription, ranging from Basic (€ 479 per server per year) to Platinum (€ 3,999 per server per year). Platinum subscription includes 24x7 support access, 30 minute emergency response time, consultative support etc. (http://www.mysql.com/products/enterprise/features.html, accessed June 2008)

(of several years) between the first download and becoming a paying customer. However, a frequent trigger for becoming a commercial MySQL customer is the use of MySQL for business critical purposes.

Lesson 7: Link the power of market and communities

In addition to providing candidates for viral marketing and product testing, communities also serve as recruiting ground for commercial customers. Seen this way, winning through incremental innovation means combining the power of community and markets, supplying both “Community” and “Enterprise” versions of the database software. MySQL exploits the power of the market by offering enterprise customers extra value for their money (e.g. new features, more functionality, better performance, and fewer errors), while community users – as previously mentioned – provide very valuable testing, error-reporting, and bug-fixing. The most important difference between the Enterprise and the Community version is *not* the software but the amount of support, customers receive from MySQL. As Kaj Arnö describes the key features of the Enterprise edition:

> *The* ***MySQL Enterprise Server*** *is for the non-DIY [do it yourself] commercial user; part of the ‘MySQL Enterprise’ subscription offering; for those who want extra help developing, deploying and managing MySQL DBs; coupled with access to MySQL technical support; assisted by new automated DBA monitoring and advisory services.* And he continues, *The driving force behind many commercial relationships involving both normal MySQL Server and MySQL Cluster is the mission-criticality of the use. If lots of customer revenues depend on MySQL working, then the commercial relationship gets going.*

By combining the power of the market with the power of the community, incremental innovations pursued simultaneously can become the kernel of sustained competitive success.

5. Discussion and conclusions

Incumbents in most industries (e.g. IBM, Oracle, Microsoft in the increasingly commoditized database market), usually win against new entrants (such as MySQL) if entrants compete on incremental innovation. MySQL provides and interesting exception to this rule: It succeeded not only to establish itself in a highly competitive market but also directed its value creating evolution into a beneficial acquisition by Sun in the spring of 2008. This was achieved through a series of incremental and software innovations, accompanied by a series of competitive as well as collaborative actions: (1) piggybacking on existing standards; (2) pursuing a lean product development strategy, distinguishing nice-to-have from need-to-have features; (3) seeking supply side economies of scale by coupling the development processes of commercial and community versions of the product; (4) actively nurturing communities; (5) using them to reduce product testing and marketing expenses; (6) applying a dual licensing strategy ; and finally (7) by combining all these actions into a coherent action portfolio, a business model unleashing the power of both community and markets.

Alternative explanations available in the current literature only partially account for the ‘winning-through-incremental-innovation’ phenomenon observed in our case. If an innovation is *competence-destroying* to industry incumbents, a radical product discontinuity or technological shift either creates a new product class (e.g. airlines vs. automobiles) or substitutes for an existing one (e.g. records versus compact disks).

However, in the case of MySQL neither a radical product discontinuity nor a substantial technological shift was present from a software development perspective.

As an alternative explanation, the phenomenon of *disruptive innovation* (Christensen, 2003) could be applied. It occurs when neglected market segments are discovered and served, based on incremental innovation that simplifies product offers in technologically over-served markets. When such market niches exhibit greater growth rates than incumbents' market do, firms populating the niche markets may win in the long run. While this explanation goes some way to explain the initial success of MySQL in the web-enabled enterprise market, the software solution offered by MySQL is neither substantially inferior to possibly over-complicated competing products, nor does the product offering serve a more rapidly growing main market (e.g. web-based enterprises such as Yahoo, Google etc).

A third, alternative explanation might be that what appears to be an incremental innovation is indeed an *architectural innovation* (Henderson & Clark, 1990), a change in the way product components are assembled. If unnoticed by the incumbent, incremental architectural changes may over time undermine its competitive position. This perspective may yield a future research agenda in that a stack-based software product could be interpreted as architectural design, integrating software functionality across firm boundaries; this would amount to a possible explanation of one particular aspect of our case story: MySQL's orchestrating its offer with other LAMP stack providers.

In sum, while alternative explanations go some way to illuminate partial aspects of how firms can win through incremental innovation in commoditized markets, such as the database software market, we believe that the lessons distilled in this paper serve as a foundation for crafting an alternative theory. The lessons suggest an integrated action set on how to combine the power of the market with active utilization of communities in an evolutionary, yet incremental innovation path to create shareholder value.

To conclude, this empirical paper used the case of MySQL to explore how entrants win substantial market acceptance based on a series of incremental software innovations, supported by an open yet proprietary source software development process, and a unique web-based business model. Against all odds of a large installed customer base of competitors, and market incumbents' deep pockets and control of substantial complementary assets, our findings suggest that a series of rapid incremental innovations, stack-based complement strategies, an open-source and community-based product testing and viral marketing approach, accompanied by a dual licensing strategy, all contribute to winning through incremental innovation in increasingly commoditized database software markets. We distill seven important lessons for winning through incremental innovation to guide and stimulate future research on the relation between innovation and competitive outcomes in contested software markets.

References

Abernathy, W. J. & Clark, K. B. (1985): "Innovation: Mapping the Winds of Creative Destruction", *Research Policy,* 14: 3-22.

Abernathy, W. J. & Utterback, J. M. (1978): *Strategic Management of Technology and Innovation*, McGraw-Hill: New York.

Adner, R. & Levinthal, D. (2001): "Demand Heterogeneity and Technology Evolution", *Management Science,* 47(5): 611–628.

Axmark, D. & Widenius, M. (1999): "MySQL Introduction", *Linux Journal*, (67).

Christensen, C. M. (2003): *The Innovator's Dilemma: The Revolutionary Book that Will Change the Way You Do Business*, HarperCollins.

Codd, E. F. (1970): "A Relational Model for Large Shared Data Banks", *Communications of the ACM,* 13(6).

Dixit, A. K. & Nalebuff, B. (1993): *Thinking Strategically: The Competitive Edge in Business, Politics, and Everyday Life*, W. W. Norton & Company.

Eisenhardt, K. (1989): "Building Theories from Case Study Research", *Academy of Management Review,* 14 (4): 532-550.

Henderson, R. M. & Clark, K. (1990). "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms", *Administrative Science Quarterly*, 35 (1): 9-30.

Holck, J. & Jørgensen, N. (2003): "Continuous Integration as a Means of Coordination: A Case Study of Two Open Source Projects", *Proceedings from ISD 2003*, Melbourne, Australia.

Holck, J., Pedersen, M. K. & Larsen, M. H. (2005): "Open Source Software Acquisition: Beyond the Business Case", *Proceedings from ECIS 2005*, Regensburg, Gernany.

Holck, J. & Zicari, R. V. (2006): "A Framework Analysis of Business Models for Open Source Software Products with Dual Licensing", *CBS/INF Working Paper*, No. 1, January 2007, Copenhagen Business School, Department of Informatics.

Horstmann, J. (2005): *Migration to Open Source Databases*, Diploma Thesis, Technical University of Berlin.

Klepper, S. (1996) "Entry, Exit, Growth, and Innovation over the Product Life Cycle." *American Economic Review*, 86 (3): 562-584.

Kroll, J. (2000): "1999 Readers' Choice Awards". *Linux Journal*, (69).

Kunze, M. (1998): "Let There be Light". *C'T*, December: 230.

Liebowitz, S. J. & Margolis S. E. (1998): "Network Externalities (Effects)", *New Palgrave Dictionary of Economics and the Law*, MacMillan.

Nelson, R.R. & Winter, S.G. (1982): *An Evolutionary Theory of Economic Change,* The Belknap Press of Harvard University Press.

Teece, D. J. (1982): "Towards an Economic Theory of the Multiproduct Firm", *Journal of Economic Behavior and Organization 3*: 39-63.

Teece, D. J. (1986) "Profiting from Technological Innovation: Implications for Integration, Collaboration, Licensing and Public Policy", *Research Policy*, 15(6): 285-305.

Teece, D., Pisano, G. & Shuen, A (1997): "Dynamic Capabilities and Strategic Management", *Strategic Management Journal*, 18(7): 509-533.

Tran, Y. (2005): "Industrial dynamics and leadership succession: The case of the fashion industry". *DRUID working paper.*

Tushman, M. L. & Anderson, P. (1986): "Technological Discontinuities and Organizational Environments", *Administrative Science Quarterly*, 31: 439-465.

Utterback, J. M. (1994): *Mastering the Dynamics of Innovation: How Companies Can Seize Opportunities in the Face of Technological Change*, Harvard Business School Press, Boston, MA.

Utterback, J. M. & Abernathy, W. J. (1975): "A Dynamic Model of Process and Product Innovation", *Omega,* 3(6): 639-656.